AUDIT REPORT

MAR 2023

ANALYTIX AUDIT

Security Assessment

Pepa Inu Farm

April 1, 2023

Audit Status: Pass

Audit Edition: Advance

Table of Contents

1 Assessment Summary

2 Technical Findings Summary

3 Project Overview

3.1 Main Contract Assessed

4 Smart Contract Risk Checks

5 Contract Ownership

7 KYC Check

8 Smart Contract Vulnerability Checks

8.1 Smart Contract Vulnerability Details

8.2 Smart Contract Inheritance Details

8.3 Smart Contract Privileged Functions

9 Assessment Results and Notes(Important)

10 Social Media Check(Informational)

11 Technical Findings Details

12 Disclaimer

Assessment Summary

This report has been prepared for Pepa Inu Farm on the Binance Smart Chain network. Analytix Audit provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Thorough line-by-line manual review of the entire codebase by industry experts.

Project Overview

Token Summary

Parameter	Result
Address	0x64491439D800dB00Db6540d44a4865B6ad1A14Bc
Name	Pepa Inu
Token Tracker	Pepa Inu (PEPA)
Decimals	0
Supply	0
Platform	Binance Smart Chain
compiler	v0.8.18+commit.87f61d96
Contract Name	MasterPepa
Optimization	Yes with 2000 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/0x64491439D800dB00Db6540d44a4865B6ad1A14Bc#code
Payment Tx	Corporate Account

Main Contract Assessed Contract Name

Name	Contract	Live
Pepa Inu	0x64491439D800dB00Db6540d44a4865B6ad1A14Bc	Yes

TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
MasterPepa	3e974fc33e93fc22007bd122d6599d66d44323fc	MasterPepa.sol
MasterPepa	fcab37d90398e8463c3fba93801109a46bb3846b	Ownable.sol
MasterPepa	586a4b7c629326cd95ad1fa4ec56828eb653f940	ReentrancyGuard.sol
MasterPepa	76d919f4fabac974b89715dffc96108c9940698c	draft-IERC20Permit.sol
MasterPepa	3a4c9c7e4d7bdc778a551c2601c77fc0a8043614	IERC20.sol
MasterPepa	1196a90a3326b97b2b68c2d1262cea7ed59153fb	SafeERC20.sol
MasterPepa	9922101e9897967872535fc44f664d8012e5d0d1	Address.sol
MasterPepa	37ccef725837f7fa2f3dd5446c62a9776b435b6f	Context.sol
MasterPepa	a31fa4118424e37612933c0c4032476574ff54d5	SafeMath.sol

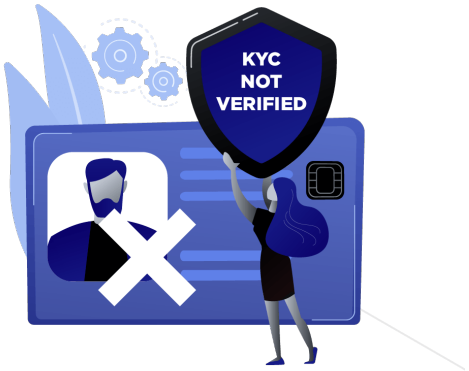
KYC Information

The Project Owners of Pepa Inu is not KYC.

KYC Information Notes:

Auditor Notes:

Project Owner Notes:



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	MasterPepa.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	MasterPepa.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	MasterPepa.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	MasterPepa.sol	L: 2 C: 0
SWC-104	Pass	Unchecked Call Return Value.	MasterPepa.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	MasterPepa.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	MasterPepa.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	MasterPepa.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	MasterPepa.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	MasterPepa.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	MasterPepa.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	MasterPepa.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	MasterPepa.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	MasterPepa.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	MasterPepa.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	MasterPepa.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	MasterPepa.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	MasterPepa.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	MasterPepa.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	MasterPepa.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	MasterPepa.sol	L: 227 C: 36
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	MasterPepa.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	MasterPepa.sol	L: 0 C: 0
SWC-123	Low	Requirement Violation.	MasterPepa.sol	L: 113 C: 9
SWC-124	Pass	Write to Arbitrary Storage Location.	MasterPepa.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	MasterPepa.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	MasterPepa.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	MasterPepa.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	MasterPepa.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	MasterPepa.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	MasterPepa.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	MasterPepa.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	MasterPepa.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	MasterPepa.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	MasterPepa.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	MasterPepa.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	MasterPepa.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-123 - Requirement Violation

CWE-573: Improper Following of Specification by Caller

Description:

The Solidity require() construct is meant to validate external inputs of a function. In most cases, such external inputs are provided by callers, but they may also be returned by callees. In the former case, we refer to them as precondition violations. Violations of a requirement can indicate one of two possible issues:

A bug exists in the contract that provided the external input.
The condition used to express the requirement is too strong.

Remediation:

If the required logical condition is too strong, it should be weakened to allow all valid external inputs.Otherwise, the bug must be in the contract that provided the external input and one should consider fixing its code by making sure no invalid inputs are provided.

References:

The use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM

Smart Contract Advance Checks

ID	Severity	Name	Result	Status
PEPA-01	Minor	Potential Sandwich Attacks.	Pass	Not-Found
PEPA-02	Informational	Function Visibility Optimization	Pass	Not-Found
PEPA-03	Minor	Lack of Input Validation.	Pass	Not-Found
PEPA-04	Major	Centralized Risk In addLiquidity.	Pass	Not-Found
PEPA-05	Major	Missing Event Emission.	Pass	Not-Found
PEPA-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Not-Found
PEPA-07	Minor	State Variables could be Declared Constant.	Pass	Not-Found
PEPA-08	Major	Dead Code Elimination.	Pass	Not-Found
PEPA-09	Major	Third Party Dependencies.	Pass	Not Found
PEPA-10	Major	Initial Token Distribution.	Pass	Not-Found
PEPA-11	Minor	Missing Init_Hash for Pair Creation if needed.	Pass	Not-Found
PEPA-12	Major	Centralization Risks In The X Role	Fail	Pending
PEPA-13	Informational	Extra Gas Cost For User..	Pass	Not-Found
PEPA-14	Medium	Unnecessary Use Of SafeMath	Fail	Pending
PEPA-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found

ID	Severity	Name	Result	Status
PEPA-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not-Found

PEPA-12 | Centralization Risks In The Owner Role

Category	Severity	Location	Status
Centralization / Privilege	Major	MasterPepa.sol: 42, 58	Pending

Description

In the contract MasterPapa, the role Owner has authority over the following functions:
function updatePepaEmission(), to update PepaEmission per block.
function emergencyWithdrawPepaRewards(), to withdraw any token left "PEPA" in the current smart contract .
function updateBoostContract() to update BoostContract, and distributing multiplied rewards.
function add() to add any lp Pool, and get rewards in the native token "PEPA" if staked.
function set() to manage any lp Pool already existing, and change "_pid, _allocPoint, _withUpdate"
Any compromise to the Owner account may allow the hacker to take advantage of this authority.
no user funds is at risk by these privilages.

Remediation

The risk describes the current project design and potentially makes iterations to improve in the security operation
and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the
client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In
general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized
mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Project Action

Project Owner Decleared actions: After the initial setup period, ownership of the contract will be transfered to a timelock controlled by a multisig wallet

PEPA-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	Medium	MasterPepa.sol: 20, 11	Pending

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations
will automatically revert in case of integer overflow or underflow.
library SafeMath {
An implementation of SafeMath library is found.
using SafeMath for uint256;
SafeMath library is used for uint256 type in contract.
totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
poolInfo[_pid].allocPoint = _allocPoint;
emit SetPool(_pid, _allocPoint);
Note: Only a sample of 2 SafeMath library usage in this contract (out of 14) are shown above.

Remediation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the
Solidity programming language

Project Action

PepaRouter emits the same events and keeps the same function signature as PancakeRouterV2, to keep it compatible with all existing libraries and analytics.

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
Major	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Minor	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
Major	1	1	0
Medium	1	1	0
Minor	3	3	1
Informational	0	0	0
Total	5	5	1

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/pepa_inu	Pass
Other	https://www.tiktok.com/@pepainu	Pass
Website	https://pepa-inu.com/	Pass
Telegram	https://t.me/pepa_inu	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Aduit Result

Final Audit Score

Review	Score
Security Score	85
Auditor Score	85

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed

Audit Passed

Current project reviewed successfully passed audit, meeting all requirements for approval per Analytix Audit guidelines.

@FreddyCryptos
Founder & CEO

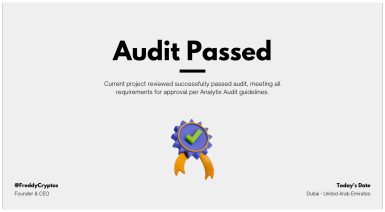
Today's Date
Dubai - United Arab Emirates

Assessment Results

Important Notes:

- No issues or vulnerabilities were found.
- We review the current farm and evaluated the current functions.
- customer adds liquidity and claim token as rewards.
- reward are paid correctly to users.

Auditor Score =85
Audit Passed



Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

Analytix Audit has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and Analytix Audit is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Analytix Audit or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Analytix Audit are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.